

Ejercicio 3. ★ Nombre los siguientes predicados auxiliares sobre secuencias de enteros:

a) pred  $????(s: \text{seq}(\mathbb{Z})) \{ (\forall i : \mathbb{Z})((0 \leq i < |s|) \rightarrow_L s[i] \geq 0)$   
 $\}$

b) pred  $????(s: \text{seq}(\mathbb{Z})) \{ (\forall i : \mathbb{Z})((0 \leq i < |s|) \rightarrow_L (\forall j : \mathbb{Z})((0 \leq j < |s| \wedge i \neq j) \rightarrow_L (s[i] \neq s[j])))$   
 $\}$

a) todos Positivos

b) no hay Repetidos

Ejercicio 4. ★ Escriba los siguientes predicados auxiliares sobre secuencias de enteros, aclarando los tipos de los parámetros que recibe:

- a)  $esPrefijo$ , que determina si una secuencia es prefijo de otra.
- b)  $estáOrdenada$ , que determina si la secuencia está ordenada de menor a mayor.
- c)  $hayUnoParQueDivideAlResto$ , que determina si hay un elemento par en la secuencia que divide a todos los otros elementos de la secuencia.
- d)  $enTresPartes$ , que determina si en la secuencia aparecen (de izquierda a derecha) primero 0s, después 1s y por último 2s. Por ejemplo  $(0, 0, 1, 1, 1, 2)$  cumple con  $enTresPartes$ , pero  $(0, 1, 3, 0)$  o  $(0, 0, 0, 1, 1)$  no. ¿Cómo modificaría la expresión para que se admitan cero apariciones de 0s, 1s y 2s (es decir, para que por ejemplo  $(0, 0, 0, 1, 1)$  o  $\langle \rangle$  sí cumplan  $enTresPartes$ )?

a) pred  $esPrefijo(s1: \text{seq}(\mathbb{T}), s2: \text{seq}(\mathbb{T})) \{$   
 $(\forall i : \mathbb{Z}) (0 \leq i < |s1| \rightarrow_L s1[i] = s2[i])$   
 $\}$

b) pred  $estaOrdenada(s: \text{seq}(\mathbb{Z})) : \text{Bool} \{$   
 $(\forall i : \mathbb{Z}) (0 \leq i < |s|-1 \rightarrow_L s[i] \leq s[i+1])$   
 $\}$

c) pred  $hayUnoParQueDivideAlResto(s: \text{seq}(\mathbb{Z})) : \text{Bool} \{$   
 $(\forall i : \mathbb{Z}) (0 \leq i < |s| \rightarrow_L (\exists x : \mathbb{Z}) (x \in s \wedge s[i] \bmod x = 0))$   
 $\}$

d) pred  $sinRepetidos(s: \text{seq}(\mathbb{Z})) : \text{Bool} \{$   
 $(\forall i : \mathbb{Z}) (0 \leq i < |s| \rightarrow_L (\forall j : \mathbb{Z}) (0 \leq j < |s| \wedge i \neq j \rightarrow_L s[i] \neq s[j]))$   
 $\}$

e) pred  $enTresPartes(s: \text{seq}(\mathbb{Z})) : \text{Bool} \{$  Revisar procedimiento  
 $(0 \leq i < |s| \wedge \text{empiezaConCeros}(s, i) \wedge \text{siguenUnos}(s, i) \wedge \text{porUltimoDos}(s, i))$   
 $\}$

pred  $\text{empiezaConCeros}(s: \text{seq}(\mathbb{Z}), i: \mathbb{Z}) : \text{Bool} \{$

$(i = 0) \rightarrow_L s[i] = 0 \vee (s[i] = 0 \wedge s[i-1] = 0 \wedge (s[i+1] = 0 \vee s[i+1] = 1))$   
 $\}$

pred  $\text{siguenUnos}(s: \text{seq}(\mathbb{Z}), i: \mathbb{Z}) : \text{Bool} \{$

$(i < |s|-1) \wedge (i > 0) \wedge s[i] = 1 \wedge [(s[i-1] = 0 \wedge s[i+1] = 1) \vee (s[i-1] = 1 \wedge (s[i+1] = 1 \vee s[i+1] = 2))]$   
 $\}$

pred  $\text{porUltimoDos}(s: \text{seq}(\mathbb{Z}), i: \mathbb{Z}) : \text{Bool} \{$

7

**Ejercicio 5.** Sea  $s$  una secuencia de elementos de tipo  $\mathbb{Z}$ . Escribir una expresión (utilizando sumatoria y productoria) tal que:

- Cuento la cantidad de veces que aparece el elemento  $e$  de tipo  $\mathbb{Z}$  en la secuencia  $s$ .
- Sume los elementos en las posiciones impares de la secuencia  $s$ .
- Sume los elementos mayores a 0 contenidos en la secuencia  $s$ .
- Sume los inversos multiplicativos ( $\frac{1}{x}$ ) de los elementos contenidos en la secuencia  $s$  distintos a 0.

a)  $\sum_{i=0}^{|s|-1} (\text{if } s[i] = e \text{ then } 1 \text{ else } 0)$

b)  $\sum_{i=0}^{|s|-1} (\text{if } (i \bmod 2 \neq 0) \text{ then } s[i] \text{ else } 0)$

c)  $\sum_{i=0}^{|s|-1} (\text{if } (s[i] > 0) \text{ then } s[i] \text{ else } 0)$

d)  $\sum_{i=0}^{|s|-1} (\text{if } (\frac{1}{s[i]} > 0) \text{ then } \frac{1}{s[i]} \text{ else } 0)$

## 2.2. Análisis de especificación

**Ejercicio 6.** Las siguientes especificaciones no son correctas. Indicar por qué y corregirlas para que describan correctamente el problema.

- a) `progresionGeometricaFactor2`: Indica si la secuencia  $l$  representa una progresión geométrica factor 2. Es decir, si cada elemento de la secuencia es el doble del elemento anterior.

```
proc progresionGeometricaFactor2 (in l: seq<Z>) : Bool
    requiere {True}
    asegura {res = True  $\leftrightarrow$  (( $\forall i : \mathbb{Z}$ ) ( $0 \leq i < |l| \rightarrow_L l[i] = 2 * l[i - 1]$ ))}  $\Rightarrow 0 < i < |l|$ 
        en  $l[0] = 2, l[0-1]$ 
        se define
```

- b) `minimo`: Devuelve en  $res$  el menor elemento de  $l$ .

```
proc minimo (in l: seq<Z>) : Z
    requiere {True}
    asegura {( $\forall y : \mathbb{Z}$ ) ( $y \in l \wedge y \neq x \rightarrow y > res$ )}
         $x$  no está declarado
```

**Ejercicio 7.** Para los siguientes problemas, dar todas las soluciones posibles a las entradas dadas:

- a) `proc indiceDelMaximo` (in  $l$ :  $seq<\mathbb{R}>$ ) :  $\mathbb{Z}$   
requiere  $\{|l| > 0\}$   
asegura  $\{0 \leq res < |l| \wedge_L ((\forall i : \mathbb{Z}) (0 \leq i < |l| \rightarrow_L l[i] \leq l[res]))\}$

- $l = \langle 1, 2, 3, 4 \rangle$   $res = \{3\}$
- $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle$   $res = \{0, 3\}$
- $l = \langle 0, 0, 0, 0, 0 \rangle$   $res = \{0, 1, 2, 3, 4, 5\}$

- b) `proc indiceDelPrimerMaximo` (in  $l$ :  $seq<\mathbb{R}>$ ) :  $\mathbb{Z}$   
requiere  $\{|l| > 0\}$   
asegura  $\{0 \leq res < |l| \wedge_L ((\forall i : \mathbb{Z}) (0 \leq i < |l| \rightarrow_L (l[i] < l[res] \vee (l[i] = l[res] \wedge i \geq res))))\}$

- $l = \langle 1, 2, 3, 4 \rangle$   $res = 3$
- $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle$   $res = 0$
- $l = \langle 0, 0, 0, 0, 0 \rangle$   $res = 0$

- c) ¿Para qué valores de entrada `indiceDelPrimerMaximo` y `indiceDelMaximo` tienen necesariamente la misma salida?

Para  $a = 1$  y  $b = 1$

**Ejercicio 8.** Sea  $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  definida como:

$$f(a, b) = \begin{cases} 2b & \text{si } a < 0 \\ b - 1 & \text{en otro caso} \end{cases}$$

Indicar cuáles de las siguientes especificaciones son correctas para el problema de calcular  $f(a, b)$ . Para aquellas que no lo son, indicar por qué.

a) proc f (in a, b: R) : R INCORRECTO

requiere {True}

asegura  $\{(a < 0 \wedge res = 2 * b) \wedge (a \geq 0 \wedge res = b - 1)\}$

*Si  $a = 2 \Rightarrow a < 0 \wedge res = 2 * b$  es falso y luego todo el asegura es falso, por más que la segunda parte sea verdadera.*

b) proc f (in a, b: R) : R CORRECTO pues en alguno de los 2 casos

requiere {True} *de verdadero siempre, ya que si a no es positivo, entra en el caso negativo del OR y viceversa.*

asegura  $\{(a < 0 \wedge res = 2 * b) \vee (a \geq 0 \wedge res = b - 1)\}$

c) proc f (in a, b: R) : R INCORRECTO pues si por ej.  $a = -1$

$a < 0 \rightarrow res = 2 * b$  es verdadero

requiere {True} pero a la vez  $a \geq 0 \rightarrow res = b - 1$

es verdadero, entonces res puede ser  $2 * b$  y  $b - 1$ ?

asegura  $\{(a < 0 \rightarrow res = 2 * b) \vee (a \geq 0 \rightarrow res = b - 1)\}$

d) proc f (in a, b: R) : R CORRECTO pues res depende de si a es positivo o negativo y cumple lo esperado

requiere {True} por la función f.

asegura  $\{res = (\text{if } a < 0 \text{ then } 2 * b \text{ else } b - 1 \text{ fi})\}$

**Ejercicio 9.** Considerar la siguiente especificación, junto con un algoritmo que dado  $x$  devuelve  $x^2$ .

proc unoMasGrande (in x: R) : R

requiere {True}

asegura  $\{res > x\}$

a) ¿Qué devuelve el algoritmo si recibe  $x = 3$ ? ¿El resultado hace verdadera la postcondición de unoMasGrande?

b) ¿Qué sucede para las entradas  $x = 0,5$ ,  $x = 1$ ,  $x = -0,2$  y  $x = -7$ ?

c) Teniendo en cuenta lo respondido en los puntos anteriores, escribir una **precondición** para unoMasGrande, de manera tal que el algoritmo cumpla con la especificación

a) El algoritmo devuelve  $9$  y cumple la postcondición de unoMasGrande

b)  $x \ x^2$  unoMasGrande c) requiere  $\{x \geq 1 \vee x < 0\}$

0,5 0,25

F

1 1

F

-0,2 0,04

V

-7 49

V

### 2.3. Relación de fuerza

Ejercicio 10. Sean  $x$  y  $r$  variables de tipo R. Considerar los siguientes predicados:

$$\begin{aligned} P1: & \{x \leq 0\} \\ P2: & \{x \leq 10\} \\ P3: & \{x \leq -10\} \end{aligned}$$

$$\begin{aligned} Q1: & \{r \geq x^2\} \\ Q2: & \{r \geq 0\} \\ Q3: & \{r = x^2\} \end{aligned}$$

a) Indicar la relación de fuerza entre P1, P2 y P3

b) Indicar la relación de fuerza entre Q1, Q2 y Q3

c) Escribir 2 programas que cumplan con la siguiente especificación:

```
proc hagoAlgo (in x: R) : R
     $\leftarrow y = x \cdot x$ 
    requiere  $\{x \leq 0\}$   $\leftarrow y = x \cdot x + 1$ 
    asegura  $\{res \geq x^2\}$ 
```

d) Se A un algoritmo que cumple con la especificación del ítem anterior. Decidir si necesariamente cumple las siguientes especificaciones:

- a) requiere  $\{x \leq -10\}$ , asegura  $\{r \geq x^2\}$  No porque  $P_1 \not\rightarrow P_3$
- b) requiere  $\{x \leq 10\}$ , asegura  $\{r \geq x^2\}$  Sí porque  $P_1 \rightarrow P_2$
- c) requiere  $\{x \leq 0\}$ , asegura  $\{r \geq 0\}$  Sí porque  $Q_1 \rightarrow Q_2$
- d) requiere  $\{x \leq 0\}$ , asegura  $\{r = x^2\}$  NO porque  $Q_1 \not\rightarrow Q_3$
- e) requiere  $\{x \leq -10\}$ , asegura  $\{r \geq 0\}$  Sí porque  $Q_1 \rightarrow Q_2$
- f) requiere  $\{x \leq 10\}$ , asegura  $\{r = x^2\}$  NO porque  $Q_1 \not\rightarrow Q_3$

e) ¿Qué conclusión pueden sacar? ¿Qué debe cumplirse con respecto a las precondiciones y postcondiciones para que sea seguro reemplazar la especificación? Dependiendo de la relación de fuerza, la pre y post condición por la que quiero reemplazar tiene que ser igual o más débil

**a es más fuerte que b si  $a \rightarrow b$  es una tautología.**

$P_3 \rightarrow P_1$  es una tautología  
si  $x \leq -10 \rightarrow x \leq 10$  siempre

$r \geq x^2 \rightarrow r \geq 0 \quad Q_1 \rightarrow Q_2$   
 $r \geq 0 \rightarrow r \geq x^2$  no es tautología  
 $r = x^2 \rightarrow r \geq x^2$  tautología

- a)  $P_1$  es más fuerte que  $P_2$  y  $P_3$  es más fuerte que  $P_1$
- b)  $Q_3$  es más fuerte que  $Q_1$  y  $Q_1$  es más fuerte que  $Q_2$

$P_3 \rightarrow P_1 \rightarrow P_2$   
 $Q_3 \rightarrow Q_1 \rightarrow Q_2$

Ejercicio 11. Considerar las siguientes dos especificaciones, junto con un algoritmo  $a$  que satisface la especificación de p2.

proc p1 (in x: R, in n: Z) : Z  
requiere  $\{x \neq 0\}$   
asegura  $\{x^n - 1 < res \leq x^n\}$  con  $x \neq 0 \rightarrow$  el resultado está entre  $x^n$  y su anterior

proc p2 (in x: R, in n: Z) : Z  
requiere  $\{n \leq 0 \rightarrow x \neq 0\}$   
asegura  $\{res = \lfloor x^n \rfloor\}$  si  $n \leq 0$  entonces  $x \neq 0$  y res es  $\lfloor x^n \rfloor$

- a) Dados valores de  $x$  y  $n$  que hacen verdadera la precondición de p1, demostrar que hacen también verdadera la precondición de p2.
- b) Ahora, dados estos valores de  $x$  y  $n$ , supongamos que se ejecuta  $a$ : llegamos a un valor de  $res$  que hace verdadera la postcondición de p2. ¿Será también verdadera la postcondición de p1 con este valor de  $res$ ?
- c) ¿Podemos concluir que  $a$  satisface la especificación de p1?

a) en p1 tendrían que elegir un  $n \leq 0$  primero...

Esto vale porque

supongo  $x = 2$  y  $n = -1$  :  $x^{-1} < res < x^0$

la precondición de p2 implica la de p1:

$$\frac{1}{2} - 1 < res < \frac{1}{2}$$

$$-\frac{1}{2} < res < \frac{1}{2}$$

$$(n \leq 0 \rightarrow x \neq 0) \rightarrow x \neq 0$$

b) si :  $n = -1$  y  $x = 2$  :  $res = \lfloor 2^{-1} \rfloor$

$$res = 0 \text{ en p2}$$

luego en p1: 0 cumple la postcondición pues  $-\frac{1}{2} < 0 < \frac{1}{2}$

sí porque  $\lfloor x^n \rfloor \rightarrow (x^{n-1} < res \leq x^n)$  pues el floor de un número cualquiera siempre está entre sí mismo y el elemento anterior

c) sí porque  $\lfloor x^n \rfloor$  es siempre menor o igual a  $x^n$  (como pide la postcond. de p1) y además siempre la parte entera de un número es mayor que su predecesor. Eso es porque tanto la pre y post condición de p2

son más fuertes que la pre y post de p2.

#### 2.4. Especificación de problemas

Ejercicio 12. Especificar los siguientes problemas:

- Dado un entero, decidir si es par
- Dado un entero  $n$  y otro  $m$ , decidir si  $n$  es un múltiplo de  $m$
- Dado un entero, listar todos sus divisores positivos (sin duplicados)
- Dado un entero positivo, obtener su descomposición en factores primos. Devolver una secuencia de tuplas  $(p, e)$ , donde  $p$  es un factor primo y  $e$  es su exponente, ordenada en forma creciente con respecto a  $p$

a) proc `pred esPar(n: Z) {`  
`res=True ↔ (n mod 2)=0 }`

proc `decidir SiEsPar {in n: Z? :Bool{`  
`requiere { n≥0}`  
`asegura { result=true ↔ esPar(n)}`  
`}`

b) proc `pred esMultiplo(n: Z, m: Z) {`  
`n mod m = 0`  
`}`

proc `esMultiploDef { in n: Z, in m: Z? :Bool{`  
`requiere { m≠0}`  
`asegura { res = true ↔ esMultiplo(n, m)}`  
`}`

c) proc `obtenerDivisores(n: Z) : seq<Z> {`

`requiere {True}`

`asegura { (forall d: Z) (d ∈ res ↔ (0 < d < n ∧ n mod d = 0) ∧ noHayRepetidos(res)) }`

d) proc `descomposicion(n: Z) : seq<(Z, Z)> {`

`requiere { n>0 }`  
`asegura { (n=1 → res=[(1, 0)]) ∨`  
`(forall i: Z) (0 ≤ i < lrest → [esPrimo(res[i][0]) ∧ (n = (prod(i+1 to lrest) res[i][0])]`  
`∧ estanOrdenados(res)) } }`

$$(n = \prod_{i=1}^{lrest-1} res[i][0] \wedge res[i][1])$$

Ejercicio 13. Especificar los siguientes problemas sobre secuencias:

- Dadas dos secuencias  $s$  y  $t$ , decidir si  $s$  está *incluida* en  $t$ , es decir, si todos los elementos de  $s$  aparecen en  $t$  en igual o mayor cantidad
- Dadas dos secuencias  $s$  y  $t$ , devolver su *intersección*, es decir, una secuencia con todos los elementos que aparecen en ambas. Si un mismo elemento tiene repetidos, la secuencia retornada debe contener la cantidad mínima de apariciones del elemento en  $s$  y en  $t$
- Dada una secuencia de números enteros, devolver aquel que dividida a más elementos de la secuencia. El elemento tiene que pertenecer a la secuencia original. Si existe más de un elemento que cumple esta propiedad, devolver alguno de ellos
- Dada una secuencia de secuencias de enteros  $l$ , devolver una secuencia de  $l$  que contenga el máximo valor. Por ejemplo, si  $l = \langle \langle 2, 3, 5 \rangle, \langle 8, 1 \rangle, \langle 2, 8, 4, 3 \rangle \rangle$ , devolver  $\langle 8, 1 \rangle$  o  $\langle 2, 8, 4, 3 \rangle$
- Dada una secuencia  $l$  con todos sus elementos distintos, devolver la secuencia de *partes*, es decir, la secuencia de todas las secuencias incluidas en  $l$ , cada una con sus elementos en el mismo orden en que aparecen en  $l$

a) proc `estaIncluida (in s: seq<T>, in t: seq<T>) : Bool{`

`requiere {True}`

`asegura { res=true ↔ (forall i: Z) ((0 ≤ i < l) ∧ (s[i] ∈ t)) }`

b) proc `interseccion (in s: seq<T>, in t: seq<T>): seq<T> {`

`requiere {True}`

`asegura { (forall i: Z) { 0 ≤ i < lrest ∧ res[i] ∈ s ∧ res[i] ∈ t ∧ (#Apariciones(res[i], res)=1) }`

$\text{aux } \# \text{Apariciones}(e, s, res) = \sum_{i=0}^{|s|-1} \text{if } s[i] = e \text{ then } 1 \text{ else } 0 \text{ fi}$

- c) Dada una secuencia de números enteros, devolver aquel que divide a más elementos de la secuencia. El elemento tiene que pertenecer a la secuencia original. Si existe más de un elemento que cumple esta propiedad, devolver alguno de ellos

proc elQueMasDivide (in  $s : \text{seq}(\mathbb{Z})$ ) :  $\mathbb{Z}$  {

requiere { True }

asegura {  $res = \sum_{i=0}^{|s|-1} \text{if masDivide}(s[i]) \text{ then } s[i] \text{ else } 0 \text{ fi}$  }

pred masDivide ( $e, s$ ) :

$(\forall p : \mathbb{Z})(p \in s \wedge p \neq e \rightarrow \text{aCuantosDivide}(e) > \text{aCuantosDivide}(p))$

↳ ¿Este decide si  $e$  divide a más que cualquier otro?

aux aCuantosDivide ( $n, s$ ) =

$\sum_{i=0}^{|s|-1} \text{if } s[i] \bmod n = 0 \text{ then } 1 \text{ else } 0$

- d) Dada una secuencia de secuencias de enteros  $l$ , devolver una secuencia de  $l$  que contenga el máximo valor. Por ejemplo, si  $l = \langle \langle 2, 3, 5 \rangle, \langle 8, 1 \rangle, \langle 2, 8, 4, 3 \rangle \rangle$ , devolver  $\langle 8, 1 \rangle$  o  $\langle 2, 8, 4, 3 \rangle$

proc secuenciaConMaximo (in  $s : \text{seq}(\text{seq}(\mathbb{Z}))$ ) :  $\text{seq}(\mathbb{Z})$  {

requiere { True }

asegura {  $(\forall i : \mathbb{Z})(0 \leq i < |s| \wedge \text{contieneElMax}(s[i], s) \rightarrow res = s[i])$  }

pred contieneElMax ( $e : \text{seq}(\mathbb{Z}), s : \text{seq}(\text{seq}(\mathbb{Z}))$ ) {

$(\forall i : \mathbb{Z})(0 \leq i < |s| \wedge (s[i] \neq e \rightarrow \text{maximo}(e) > \text{maximo}(s[i]))$

}

aux maximo ( $s : \text{seq}(\mathbb{Z})$ ) =

$\sum_{i=0}^{|s|-1} \text{if } (\text{esMaximo}(s[i], s) \text{ then } s[i] \text{ else } 0 \text{ fi}$

pred esMaximo ( $e : \mathbb{Z}, s : \text{seq}(\mathbb{Z})$ ) {

$(\forall p : \mathbb{Z})(p \in s \wedge p \neq e \rightarrow e > p)$

}

consultar procedimiento

- e) Dada una secuencia  $l$  con todos sus elementos distintos, devolver la secuencia de *partes*, es decir, la secuencia de todas las secuencias incluidas en  $l$ , cada una con sus elementos en el mismo orden en que aparecen en  $l$

```
proc secuenciaDePartes(in s: seq<Z>): seq<seq<Z>> {
    requiere { ( $\forall i, j: Z$ ) ( $0 \leq i, j < |s| \wedge i \neq j \rightarrow_L s[i] \neq s[j]$ ) } → son todos distintos.
    asegura { ( $\forall i: Z$ ) ( $0 \leq i < |res| \rightarrow_L res[i] = partesDe(s, i)$ ) }
}
```

Cómo sería la combinación de elementos.

## 2.5. Especificación de problemas usando inout

**Ejercicio 14.** Dados dos enteros  $a$  y  $b$ , se necesita calcular su suma y retornarla en un entero  $c$ . ¿Cuáles de las siguientes especificaciones son correctas para este problema? Para las que no lo son, indicar por qué.

- a) proc sumar (inout a, b, c: Z)      No veo que se use a y b como inout  
 requiere {True}  
 asegura { $a + b = c$ } ←
- b) proc sumar (in a, b: Z, inout c: Z)      eval es la diferencia entre  
 requiere {True}  
 asegura { $c = a + b$ } ← poner el res = 0 = res  
 si se está asegurando la igualdad
- c) proc sumar (inout a, b: Z, inout c: Z)  
 requiere { $a = A_0 \wedge b = B_0$ }  
 asegura { $a = A_0 \wedge b = B_0 \wedge c = a + b$ }  
 No entiendo por qué se vuelve a definir la igualdad.

**Ejercicio 15.** Dada una secuencia  $l$ , se desea sacar su primer elemento y devolverlo. Decidir cuáles de estas especificaciones son correctas. Para las que no lo son, indicar por qué y justificar con ejemplos.

- a) proc tomarPrimero (inout l: seq(R)) : R X  
 requiere { $|l| > 0$ }  
 asegura { $res = head(l)$ } → debería ser head(old(l)) y ademas l = tail(old(l))
- b) proc tomarPrimero (inout l: seq(R)) : R ✓  
 requiere { $|l| > 0 \wedge l = L_0$ }  
 asegura { $res = head(L_0)$ } → necesito decir algo sobre l?
- c) proc tomarPrimero (inout l: seq(R)) : R X       $L_0 = [1, 2, 3] \quad l = [4, 3]$   
 requiere { $|l| > 0$ }  
 asegura { $res = head(L_0) \wedge |l| = |L_0| - 1$ }      res = 1 pero l?
- d) proc tomarPrimero (inout l: seq(R)) : R ✓  
 requiere { $|l| > 0 \wedge l = L_0$ }  
 asegura { $res = head(L_0) \wedge l = tail(L_0)$ }

**Ejercicio 16.** Dada una secuencia de enteros, se requiere multiplicar por 2 aquellos valores que se encuentran en posiciones pares. Indicar por qué son incorrectas las siguientes especificaciones y proponer una alternativa correcta.

- a) proc duplicarPares (inout l: seq(Z)) → no está definido qué pasa con los índices impares.  
 requiere { $l = L_0$ }  
 asegura { $|l| = |L_0| \wedge (\forall i: Z)(0 \leq i < |l| \wedge i \text{ mód } 2 = 0 \rightarrow_L l[i] = 2 * L_0[i])$ }
- b) proc duplicarPares (inout l: seq(Z)) → nota megarantiza que  $|l| = |L_0|$   
 requiere { $l = L_0$ }  
 asegura { $(\forall i: Z)((0 \leq i < |l| \wedge i \text{ mód } 2 \neq 0) \rightarrow_L l[i] = L_0[i]) \wedge (\forall i: Z)((0 \leq i < |l| \wedge i \text{ mód } 2 = 0) \rightarrow_L l[i] = 2 * L_0[i])$ }
- c) proc duplicarPares (inout l: seq(Z)) : seq(Z)  
 requiere {True}  
 asegura { $|l| = |res| \wedge (\forall i: Z)((0 \leq i < |l| \wedge i \text{ mód } 2 \neq 0) \rightarrow_L res[i] = l[i]) \wedge (\forall i: Z)((0 \leq i < |l| \wedge i \text{ mód } 2 = 0) \rightarrow_L res[i] = 2 * l[i])$ } → debería usar old(l)[i] para determinar el valor de los res[i]

Ejercicio 17. Especificar los siguientes problemas de modificación de secuencias:

- proc primosHermanos(inout  $l : \text{seq}(\mathbb{Z})$ ), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si  $l = \langle 6, 5, 9, 14 \rangle$ , luego de aplicar  $\text{primosHermanos}(l)$ ,  $l = \langle 5, 3, 7, 13 \rangle$
- proc reemplazar(inout  $l : \text{seq}(\text{Char})$ , in  $a, b : \text{Char}$ ), que reemplaza todas las apariciones de  $a$  en  $l$  por  $b$
- proc limpiarDuplicados(inout  $l : \text{seq}(\text{Char})$ , out  $dup : \text{seq}(\text{Char})$ ), que elimina los elementos duplicados de  $l$  dejando sólo su primera aparición (en el orden original). Devuelve además,  $dup$  una secuencia con todas las apariciones eliminadas (en cualquier orden)

a) proc primosHermanos(inout  $l : \text{seq}(\mathbb{Z})$ ) :  $\text{seq}(\mathbb{Z})$  {

requiere:  $\{ |l| > 0 \wedge l = l_0 \wedge (\forall i : \mathbb{Z}) (0 \leq i < |l| \wedge l[i] > 2) \}$

asegura:  $\{ (\forall i : \mathbb{Z}) (0 \leq i < |l| \wedge l[i] = \text{obtenerPrimoAnterior}(\text{old}(l)[i])) \}$

pred esElPrimoAnterior ( $p : \mathbb{Z}$ ,  $n : \mathbb{Z}$ ) {

}

aux obtenerElPrimoAnterior ( $n : \mathbb{Z}$ ) {

$\sum_{i=2}^n (\text{if esElPrimoAnterior}(i, n) \text{ then } i \text{ else } 0) \text{ fi }$  }

cómo sé que es  
el inmediatamente  
anterior?

- proc reemplazar(inout  $l : \text{seq}(\text{Char})$ , in  $a, b : \text{Char}$ ), que reemplaza todas las apariciones de  $a$  en  $l$  por  $b$

proc reemplazar (inout  $l : \text{seq}(\text{char})$ , in  $a, b : \text{char}$ , out  $dup : \text{seq}(\text{char})$ ) {

requiere { True }

asegura:  $\{ (\forall i : \mathbb{Z}) (0 \leq i < |l| \wedge l[i] = (\text{if old}(l)[i] = a \text{ then } b \text{ else } \text{old}(l)[i])) \}$

}

- proc limpiarDuplicados(inout  $l : \text{seq}(\text{Char})$ , out  $dup : \text{seq}(\text{Char})$ ), que elimina los elementos duplicados de  $l$  dejando sólo su primera aparición (en el orden original). Devuelve además,  $dup$  una secuencia con todas las apariciones eliminadas (en cualquier orden)

proc limpiarDuplicados (inout  $l : \text{seq}(\text{Char})$ , out  $dup : \text{seq}(\text{Char})$ ) {

requiere { True }

asegura:  $\{ (\forall i : \mathbb{Z}) (0 \leq i < |\text{old}(l)| \rightarrow_l l[i] =$

como hago para quedarme con  
la primera aparición?

$(\forall j : \mathbb{Z}) (0 \leq j < |\text{dup}| \rightarrow_l \text{dup}[i] = \text{aparicionesRepetidasDeEle?})$